Projet C++ L2 2015
Aide à l'optimisation du planning dans une grande école d'ingénieurs en informatique

Franck Lepoivre

v1, 7 mai 2012

Ceci est la version finale qui apporte les dernière précisions. Vous êtes invités à notifier sans délai vos équipes à vos délégués qui vont organiser les ordres de passage en servant au mieux les premiers à se manifester (et inversement).

1 Spécifications

Réaliser un programme en C++ qui permet d'appuyer l'optimisation du planning dans une école d'ingénieurs en informatique.

1.1 Principe

Produire le planning consiste à produire un agenda annuel, semaine par semaine, qui définit l'ensemble des rencontres pédagogiques entre les enseignants et les élèves au cours desquelles sont transmis savoir et savoir-faire. Pour définir cette chronologie de rencontres, il faut tenir compte :

- de l'ensemble des salles (*plan de l'école*) et de leur disponibilité (ou non)
- de l'ensemble des enseignements (*cursus*)
- de l'ensemble des élèves (*promos*, *groupes*)
- de l'ensemble des enseignants, de leurs charges de cours et de leur disponibilité (ou non)
- de l'ensemble des contraintes d'agenda (jours travaillés ou non, périodes de vacances, horaires de travail, exception et condition de leur exercice, etc)
- de l'ensemble des contraintes d'association et d'exclusions mutuelles que votre bon sens vous permettra de lister (par exemple : non ubiquité).
- etc.

La complexité du système de contraintes rendent difficile la production d'un planning qui maximise l'efficacité pédagogique en minimisant les coûts d'opérations (coût et efficacité à comprendre collectivement et individuellement). D'autre part, le manque de visibilité sur les contraintes à terme rend difficile l'affectation d'un agenda au delà d'un certain horizon (de temps). Notamment, il faut prévoir de pouvoir s'adapter à l'évolution des contraintes, ce qui suppose un système qui optimise en temps réel le planning à venir en intégrant ces variations de contraintes.

1.2 Front office

Vous proposerez une solution interactive qui permet :

- La saisie des contraintes générales (cursus, salles, etc) et propres à chacun des acteurs impliqués (disponibilités personnelles, etc),

– Le calcul de l'affectation partielle[1] et de la solution optimale à l'instant t,
– La présentation de la vue de son planning individuel à chacun des acteurs, en vue de sa validation ou sa réfutation,
– La proposition instantanée de solutions alternatives, quand une proposition est réfutée.

1.3 Back office : PPC et CSP

Vous vous intéresserez à la modélisation et la résolution de la problématique d'optimisation sous l'angle de la programmation par contraintes.

2 Consignes

2.1 Équipes

Travail impérativement par équipe de deux.
Une demande de dérogation pour raison autre que d'arithmétique modulaire serait déplacée.
Votre équipe et vos préférences d'ordre de passage doivent être notifiés à votre délégué sans délai. Celui-ci favorisera naturellement les premiers à se manifester.

2.2 Date limite de livraison

La livraison de votre travail doit être effectuée au plus tard le 30 mai à 23h59.
Le groupe PL seul pourra donc effectuer quelques améliorations post-soutenance.

2.3 Mode de livraison

Attention, une seule livraison par équipe !
Livraison par email à l'adresse indiquée ci-dessous. Vous recevrez un accusé de réception (prévoir un délai de 2 à 3 jours). Si au-delà de ce délai vous n'avez pas reçu cet accusé de réception, il vous appartient de vous en inquiéter.

Adresse Pepper Labs Livraisons <plabs.livraisons@gmail.com>[2].

Objet EFREI L2 2015 PC++ *NOM1 - NOM1*.

Attachement Un fichier d'archive nommé EFREI L2 PC++ *NOM1 - NOM1.EXT*(RAR|ZIP|...) qui contient votre livrable tel que défini ci-après.

2.4 Forme et contenu du livrable

Votre livrable se présente sous la forme d'une archive. Il doit respecter les contraintes de *packaging* énoncées ci-dessous. En particulier, il ne doit contenir aucun fichier exécutable ni aucun fichier résultant de la compilation.

1. Vocable PPC.
2. Attention, cette adresse est réservée à la livraison. Pour toute communication autre (une question par exemple), utiliser l'adresse franck.lepoivre@gmail.com.

2.4.1 Structure

A la racine .

1. Un script de compilation pour Windows, qui utilise `gcc` (supposé rendu accessible par la variable système `PATH`), ou bien un fichier de projet Codeblocks *`votre_projet`*`.cbp` qui permet de compiler votre programme indépendamment de la position absolue de l'emplacement où est décompressée votre archive.
2. Un fichier `LISEZMOI.txt` dont le contenu est spécifié ci-dessous.

Un répertoire `src` qui contient l'ensemble de vos fichiers sources, éventuellement distribués sur plusieurs sous-répertoires.

Un répertoire `lib` qui contient l'ensemble des librairies tierces intégrées par l'édition de liens.

Un répertoire `doc` qui contient l'ensemble de la documentation demandée, idéalement au format PDF.

Un répertoire `test` qui contient un jeu de tests avec une distinction claire entre les entrées et les sorties. Ceux-ci pourront, outre valider le bon fonctionnement de votre solution, permettre d'illustrer votre *manuel utilisateur*.

2.4.2 Documentation

La documentation comprend :

Un dossier de conception de votre application, qui synthétise les fonctionnalités sur environ 5 pages, puis présente l'architecture et le fonctionnement de votre solution à l'aide de diagrammes de classes et de Design Patterns (en présenter au moins deux) sur environ 15 pages.

Un manuel d'utilisation court et intuitif, idéalement intégré dans votre application sous forme d'aide.

Un manuel d'installation et de paramétrage, si vous estimez que la complexité de votre application le nécessite. Vous pouvez sinon vous contenter d'une rubrique de quelques lignes à ce sujet dans le fichier `LISEZMOI.txt`.

LISEZMOI.txt

Votre fichier `LISEZMOI.txt` comprend impérativement deux sections :

- Un état d'avancement honnête et fiable de vos travaux, en 10 lignes maximum.
- Les références aux sources qui ont pu inspirer votre conception (Livres, sites Internet), les éventuels élèves avec qui vous avez partagé des idées et dont le code pourrait ressembler au votre (l'échange de code est formellement proscrite : triche $\Rightarrow$ 0).)
- En option, une mise en avant de fonctionnalités supplémentaires que vous avez pris l'initiative de réaliser, tout ce qui vous semble important de souligner pour vous distinguer des autres équipes.

3 A propos de l'évaluation

3.1 Soutenance

Les créneaux de soutenances suivants viennent d'être attribués :

Groupe A le vendredi 1er juin, de 8h à 13h.

Groupe B le vendredi 1er juin, de 14h à 19h.

Groupe C [HYPOTHESE : A CONF] le samedi 2 juin, de 8h à 13h.

Groupe D le jeudi 31 mai, de 8h à 13h.

Groupe PL le vendredi 25 mai, de 8h30 à 12h.

3.2 Culture du résultat

Un programme qui comporte des milliers de lignes de code mais qui ne fonctionne pas ne vaut pas grand chose. Il vaut mieux sécuriser votre affaire en avançant par petites étapes bien assurées [3] que de foncer tête baissée dans l'écriture de code C++.

3.3 Qualité documentaire

La documentation est évaluée hors soutenance et vient compléter la construction de la note EXE + CODE + COM en soutenance. La qualité éditoriale n'est clairement pas affaire de quantité.

3.4 Professionnalisme

Comme il va de soi, il n'est pas évalué à proprement parler. Il s'agit seulement de pénalités qui s'appliquent dès lors que vous ne respectez pas les consignes.

Par exemple, ne pas fournir le livrable sous la forme demandée (Mauvais objet pour l'email, mauvais nom pour votre fichier source, programme en plusieurs fichiers s'il est demandé de n'en fournir qu'un, etc.) peut vous valoir jusqu'à 3 pts de pénalités, en particulier si vous cumulez.

Plus grave : vous ne respectez pas la *deadline* : 3 pts de pénalité par jour de retard. [4].

Encore plus grave : vous trichez en fournissant le code d'autrui sans y faire référence dans le but d'être gratifié(e) d'une note sans rapport avec votre niveau réel, c'est à dire illégitime ⇒ zéro + convocation pour demande d'explication.

3. Le programme compile, s'exécute sans planter et fait bien ce qu'on attend de lui.
4. Au *prorata temporis*. Et 1 pt de bonus par jour d'avance, pour un maximum de 2 pt.

4 Annexe

4.1 Annoncé dans la version RFC du 5 avril

. . . à venir prochainement (vers le 15 avril) : le meilleur dossier de spécifications fonctionnelles et de conception + interfaces de la part des M1 : c'est leur projet en cours de COO avancée avec UML et les Design Patterns. Les équipes qui suivront scrupuleusement ce CDC pourront creuser l'écart avec qui se contente de viser la moyenne et quelques.

4.2 Suite à livraison tardive, le 7 mai, de ce complément

Vu que nous avons tardé à fournir le complément M1, il serait mal venu de vous pénaliser sur ce point, en particulier celles et ceux qui auront travaillé suffisamment en amont.

Vous pourrez creuser l'écart sur la base du meilleur dossier publié aujourd'hui sur campus, en envisageant des ajustements et extensions fonctionnelles de ce que vous avez déjà réalisé, et sans remettre en question votre modèle.

La valeur de votre solution tient bien davantage à la fonctionnalité et l'ergonomie générale qu'au modèle de données sous-jacent.

4.3 A propos du meilleur dossier M1

Les M1 (IF) devaient réaliser un travail d'analyse et de conception d'un système amélioré de gestion du planning au sein d'une grande école (toute ressemblance avec une situation réelle est purement fortuite).

Les élèves ont travaillé par équipes de trois pour conduire la synthèse fonctionnelle de l'existant, l'analyse des besoins, puis la conception avancée (avec le langage de modélisation UML pour support) d'une solution qui réponde à ces besoins. Ce travail s'achève sur la fourniture d'interfaces que peuvent réaliser les L2 qui le souhaitent, dans le cadre de leur projet en cours de C++.

Ce meilleur travail a été réalisé par l'équipe Claudia COLQUE, Juan EVANGELISTA, Andry RATSARAZAKA. Ils ont tous les trois donné leur accord pour la diffusion de leur dossier d'analyse et de conception auprès des L2, dans le cadre de leur sujet de projet C++ (même thématique, mais présentée sous l'angle plus technique d'un problème de satisfaction de contraintes).

L'archive ci-jointe comprend :

- le sujet (3 pages),
- le dossier d'analyse [fonctionnelle] (10 pages),
- le dossier de conception [technique] (20 pages),
- le jeu d'interfaces à réaliser par la MOE (30 interfaces Java).